Project 2 - Working with data structures

Project motivation

In the first project, we tried working with text and a limited spreadsheet. The main limitation was the sequential modification of the input data. In this project we will want to create a spreadsheet that can work with data throughout the table. Additionally, the spreadsheet program will not have a limit on the contents of the cells or the number of cells in the table.

Project description

The aim of the project is to create a program that will implement the basic operations of spreadsheets. The input of the program will be the text data of the table, the input of operations will be through command line arguments and the result of the table processing will be saved by the program in the specified file.

Detailed specifications

Implement the program in the source file *sps.c* (SPreadSheet). The input data will be read from a file whose name will be entered on the command line. The program performs operations specified by "spreadsheet commands", which it performs in the sequence specified on the command line.

Translation and submission of the source file

Upload: Upload the source file *sps.c* through the information system.

Translation: Compile the program with the following arguments

```
$ gcc -std = c99 -Wall -Wextra -Werror sps.c -o sps
```

Execution syntax

The program starts in the following form: (./sps indicates the location and name of the program):

```
. / sps [-d DELIM] CMD_SEQUENCE FILE
```

- Argument -d specifies which characters can be interpreted as delimiters for individual cells. Each character in the input line that is contained in a DELIM string acts as a delimiter for two adjacent cells. By default, DELIM is a string containing a space. Multiple occurrences of the same characters in a DELIM string are ignored. The first character from the DELIM string will also be used as a delimiter for output values. DELIM must not contain quotation marks or a backslash.The -d DELIM argument is optional.
- Argument CMD_SEQUENCE is one argument containing a sequence of commands. Multiple spreadsheet commands are separated by a semicolon. The command must not be empty.
- Argument FILLET specifies the name of the table file.

- Optional (facilitates advanced project development, no need to obtain a score): Commands can be saved in a text file. Each command will be separated by an end-of-line character. The command file will be specified instead of the command sequence as an argument: -cCOMMAND_FILE.

Table format

A table is a text file containing rows of a table (each row is terminated by an end-of-line character). Each line contains individual cells that are separated by just one character from the delimiter set. The data in the cell is text (integers and floating-point numbers are in decimal and are compatible with C notation). Text strings can be enclosed in quotation marks (at the beginning and at the end of the cell), they can also contain introductory characters (so-called Escape sequences) using a backslash: \X represents the character X, where X can be any character (except the end-of-line character).

Spreadsheet commands

The spreadsheet command always applies to selected cells. The selected cells can be either individual cells or an entire row, an entire column or a so-called cell window. Each cell is identified by a row and column index, counting from 1. The initial selection is the first cell in the table (row 1, column 1). The new cell selection command always resets the previous selection. Different commands can be executed for a given selection. Commands differ depending on whether they work on individual cells (in this case the command is applied sequentially line by line and in each row on columns - ie as we are used to reading the text), whether they work on the whole column (then they are applied only once for each column of the selection, with an increasing index), or whether they work over an entire row (then they are applied only once for each row of the selection, with an increasing index).

Selected commands can accept a parameter as a string called STR. Such a string is subject to the same rules as the text content of the cell specified in the paragraph describing the format of the table (ie it can be in quotation marks, it can contain special characters preceded by a backslash).

Commands for changing the selection

The selection of cells can exceed the boundaries of the input table. In this case, the table will be expanded with the required number of rows and columns containing empty cells until the current selection.

- [R, C] - cell selection on row R and column C.
- [R, _] - selection of the whole line R.
- [_, C] - selection of the whole column C.
- [R1, C1, R2, C2] - selection of the window, ie all cells on row R and column C, for which R1 <= R <= R2, C1 <= C <= C2. If instead of the number R2 resp. C2 will be a hyphen, thus replacing the maximum line resp. column in the table.
- [_, _] - selection of the whole table.
- [min] - in an existing cell selection, it finds the cell with the minimum numeric value and sets the selection to it.
- [max] - similar to the previous command, but finds the cell with the maximum value.
- [find STR] - in an existing cell selection, selects the first cell whose value contains the substring STR.
- [_] - resets the selection from the temporary variable (see below).

Commands for editing the table structure

- irow - inserts one blank line to the left of the selected cells above the selected cells.
- arow - adds one blank line to the right of the selected cells below the selected cells.
- drow - deletes the selected rows.
- icol - inserts one empty column to the left of the selected cells.
- acol - adds one empty column to the right of the selected cells.
- dcol - deletes the selected columns.

Note inserted rows and columns have an effect beyond the selected cells, adding or removing columns or rows of the entire table.

Commands for editing cell contents

- set STR - sets the cell value to the string STR. The string STR can be enclosed in quotation marks and can contain special characters preceded by a slash (see table format)
- clear - the contents of the selected cells will be deleted (the cells will have empty contents)
- swap [R, C] - replaces the contents of the selected cell with the cell in row R and column C
- sum [R, C] - stores the sum of the values of the selected cells (corresponding to the format% gu printf) in the cell on row R and column C. Selected cells without a number will be ignored (as if they were not selected).
- avg [R, C] - same as sum, but the arithmetic mean of the selected cells is stored
- count [R, C] - same as sum, but stores the number of non-empty cells from the selected cells
- only [R, C] - stores the string length of the currently selected cell in the cell on row R and column C.

Commands for working with temporary variables

The spreadsheet allows you to work with 10 so-called temporary variables identified as _0 to _9. Temporary variables have an empty value when the program is started. One temporary variable identified as an underscore _ is for cell selection (i.e., it remembers which cells were selected).

- def _X - the value of the current cell will be set to the temporary variable X (where X can be a value from 0 to 9)
- use _X - the current cell will be set to the value from the temporary variable X (where X identifies the temporary variable _0 to _9)
- inc _X - the numeric value in the temporary variable will be incremented by 1. If the temporary variable does not contain a number, the resulting value of the variable will be set to 1.
- [set] - sets the current cell selection to the temporary variable _ (only which cells are selected, not their contents)

Commands for controlling a sequence of commands

The following commands are optional and are for experimentation, inspiration, and amusement only:

- goto + N - skips N of the following commands in the command sequence (note. goto +1 is a command without effect)
- goto -N - returns in the sequence of commands by N commands (note. goto -0 is a program loop without effect).
- iszero _X + -N - skips (in the case of + N) or returns (in the case of -N) the N statements in the sequence, but only if the value of the temporary variable _X (0 <= X <= 9) is equal to 0.

- sub _X _Y - subtracts the value of the _Y variable from the value in the temporary variable _X. Stores the result in the variable _X.

Implement a command loop detector.

Exit

The output table will be modified according to the following rules:

- Each cell will be separated by the first character from the DELIM set.
- ~~If a cell contains a quotation mark or a character that matches one of the delimiters, it will be enclosed in quotation marks.~~
- ~~If a cell contains characters that would be considered special with respect to the table format, they will be preceded by a backslash.~~
- The program must not change the meaning of individual characters in cells.
- If the cell text contains quotation marks or a delimiter symbol, the entire cell text will be enclosed in quotation marks.
    - eg two cells with a space as a delimiter, in C as a field initializer {"1 "," hello world "} will be printed as one line 1 "hello world"
- If the cell text contains a quotation mark or a backslash, these will be preceded by a backslash.
    - eg cell with text: He said: "Wow!" will be converted to: "He said: \" Wow! \ ""
- Each row of the table will contain the same number of cells. The table will contain all cells that are non-empty. The table will not contain any empty last column.

Implementation details

- The maximum number of orders is 1000. The
- maximum length of one order is 1000.
- No command number identifying a row or column can be less than 1.
- If a row contains more cells than another row of the table, the other rows of the table will be filled (on the right) with empty cells so that the table is aligned, ie each row contains the same number of cells. Line alignment is performed before the first command is applied.

Implementation tips

- Implement a function to load a string with the option to be enclosed in quotation marks and containing special characters preceded by a backslash. Implement a function for listing such a string corresponding to the output format of the table.
- Define data types for cell, row, and table.
- Implement functions to load a table from a file into memory. Implement a table alignment feature. Implement a table dump function (on stdout for development and debugging purposes, without modifying the original file). Define a data type for a
- general command (command operations and parameters). Implement functions for converting a text-defined command into a structure that you define. Implement a function to convert a sequence of text-defined commands to a sequence of commands.
- Define a data type for cell selection. Define temporary variables _, _0 to _9.
- Implement a single cell selection feature. Implement functions corresponding to the selected table processing commands.

- Implement the functions of other selection methods. Implement functions that implement other commands. Implement
- an interpreter function that executes individual commands sequentially on a specified sequence of commands. Tune in,
- study behavior in a variety of situations. Verify that the memory works correctly with the valgrind tool.

- Implement control commands. Experiment with different command sequences. Look for sequences of commands that e.g.
    - add a column with numbered rows,
    - duplicate the table,
    - delete all rows except the first one, which contains the selected string, sort the
    - table by the length of the string in the first column, and so on.

Unexpected behavior

Respond to program runtime errors as usual: Respond to unexpected input data, input data format, or function call errors by interrupting the program with a brief and concise error message on the appropriate output and the corresponding return code. Reports will be in ASCII encoding in Czech or English.

Examples of inputs and outputs

```
$ cat tab.txt
Bill Gates 15,000,000
Franta Odvedle 15
Pepik Ajtak 42
```

The apostrophes in the following program run example cause the string between them to be specified as a single argument. The original example involved cell selection 1, _; irow. For structure editing commands tables, these cases will not be tested. They will only be tested on a selection of individual cells.

```
$ ./sps '[1,1]; irow; [1,1]; set First Name; [1,2]; set Last Name; [1,3]; set Plat' tab.txt $ cat tab.txt

First Name Last Name Salary
Bill Gates 15,000,000
Franta Odvedle 15
Pepik Ajtak 42
$ ./sps '[_, _]; [max]; def _0; [2,3, -, 3]; use _0' tab.txt $ cat tab.txt

First Name Last Name Salary
Bill Gates 15,000,000
Franta Odvedle 15000000 Pepik
Ajtak 15000000
```

Note: The following numbering is easier to achieve. Here is just an example for applying a command over individual cells: swap will be applied repeatedly for each selected cell, which has the side effect of moving the cells one to the right:

```
$ ./sps' [1,3]; acol; [1, _]; swap [1,4]; inc _0; [2, _]; swap [2,4]; [2,1]; use _0 ; inc _0; [3, _]; swap [3,4]; [3,1]; use _0; inc _0;
[4, _]; swap [4,4]; [4,1]; use _0 'tab.txt $ cat tab.txt
```

First Name Last Name Salary
1 Bill Gates 15000000
2 Franta Odvedle 15000000 3 Pepik
Ajtak 15000000

Evaluation

The following factors have the main influence on the final evaluation:

1. translatability of the source file,
    1. the source file must be translatable and run on GNU / Linux, for example on the server merlin.fit.vutbr.cz (Note: The code should be written in C according to the ISO C 99 standard, which should work the same on all platforms. If the code does not work on the merlin machine, it will not work on many other platforms).
2. decomposition of the problem into subproblems (suitable functions, suitable length of functions and parameters of functions),
3. correct choice of data types, creation of new types,
4. correct functionality:
    1. with a possible restriction on the size of the table (better to focus on the implementation of functionality than working with an unlimited table)
    2. selection of one cell, one row, one column
    3. commands for editing the table structure (for commands for editing the table structure, only cases over the selection of a single cell will be evaluated)
    4. commands for searching in the table, commands for aggregating calculations (sum, avg)
    5. other commands
5. treatment of error conditions.

Premium rating

Premium evaluation is voluntary and you can get bonus points for it. The condition for the award of bonus points is a successful defense of the project and excellent elaboration of the standard assignment. The final evaluation is fully in the competence of the teacher to whom the project will be defended and who will evaluate the project. The amount of premium points also depends on the sophistication of the solution. The premium implementation should include control commands.